High Low Number in CSharp

Welcome to our tutorial and blog post on creating a High-Low Number program in C#! Are you ready to dive into the world of numbers and challenge your programming skills? In this tutorial, we will walk you through the process of building a console program that determines the highest and lowest number entered by the user.

Understanding the concepts of conditional statements, user input handling, and basic mathematical operations will be key to mastering this project. By the end of this tutorial, you will have a solid grasp of how to compare and analyze numbers in C#.

Introduction

A High-Low Number Checker program is a software application that allows users to input multiple numbers and identifies the highest and lowest numbers among them. This program is designed to analyze the numerical values provided by the user and provide immediate feedback on the extremes within the set of numbers.

The purpose of a High-Low Number Checker program is to automate the process of determining the highest and lowest numbers, saving time and effort that would otherwise be spent manually comparing the values. By leveraging the power of programming, this program enables quick and accurate identification of the numerical extremes, making it useful in various scenarios.

High-Low Number Checker programs can be valuable in a range of applications. For example, in data analysis, it can help identify the highest and lowest values in a dataset, providing insights into outliers or extreme observations. In games or simulations, it can be used to keep track of the highest and lowest scores or rankings. Additionally, in general number analysis tasks, it can aid in finding the range or boundaries of a given set of numbers.

Creating a High-Low Number Checker program in C# or any other programming language not only sharpens programming skills but also deepens understanding of fundamental concepts like conditional statements, user input handling, and mathematical operations. It empowers programmers to efficiently analyze numerical data and make informed decisions based on the highest and lowest values.

By following this tutorial, you will gain hands-on experience in developing a High-Low Number Checker program in C#, enhancing your programming abilities and enabling you to tackle similar problems in various domains.

Objectives

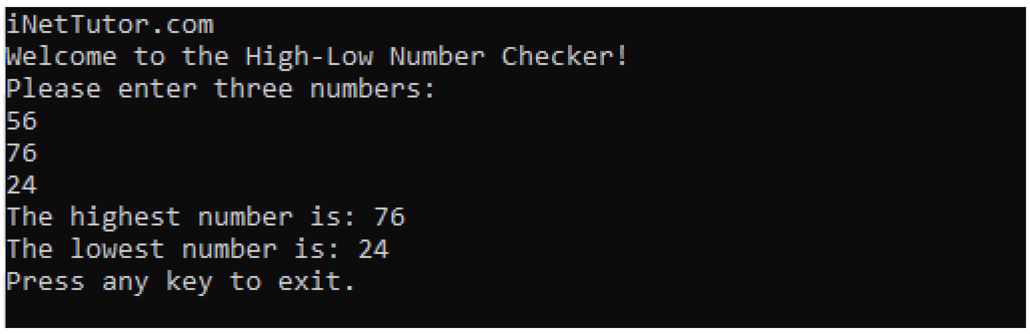
Outcome-based objectives for the lesson and topic on High-Low Number in C# can include:

1. Objective: Develop proficiency in programming logic and conditional statements.
    a. Outcome: Students will be able to apply conditional statements to compare and analyze numerical values effectively.
2. Objective: Enhance problem-solving skills through practical application.
    a. Outcome: Students will be able to devise strategies to determine the highest and lowest numbers among a set of user inputs.
3. Objective: Improve understanding of user input handling and error validation.
    a. Outcome: Students will be able to implement input validation techniques to handle invalid or unexpected user inputs.
4. Objective: Foster familiarity with mathematical operations for number comparison.
    a. Outcome: Students will be able to utilize mathematical functions and operators to identify the highest and lowest numbers.

Source code Example

```
1.  using System;
2.  
3.  namespace HighLowNumberCSharp
4.  {
5.      class Program
6.      {
7.          static void Main(string[] args)
8.          {
9.              Console.WriteLine("iNetTutor.com");
10.             Console.WriteLine("Welcome to the High-Low Number Checker!");
11.             Console.WriteLine("Please enter three numbers:");
12. 
13.             // Read user input as strings
14.             string input1 = Console.ReadLine();
15.             string input2 = Console.ReadLine();
16.             string input3 = Console.ReadLine();
17. 
18.             // Convert the input strings to integers
19.             int number1, number2, number3;
20.             bool isNumber1Valid = int.TryParse(input1, out number1);
21.             bool isNumber2Valid = int.TryParse(input2, out number2);
22.             bool isNumber3Valid = int.TryParse(input3, out number3);
23. 
24.             if (isNumber1Valid && isNumber2Valid && isNumber3Valid)
25.             {
26.                 int highest = Math.Max(Math.Max(number1, number2), number3);
27.                 int lowest = Math.Min(Math.Min(number1, number2), number3);
28. 
29.                 Console.WriteLine($"The highest number is: {highest}");
30.                 Console.WriteLine($"The lowest number is: {lowest}");
31.             }
32.             else
33.             {
34.                 Console.WriteLine("Invalid input. Please enter valid numbers.");
35.             }
36. 
37.             Console.WriteLine("Press any key to exit.");
38.             Console.ReadKey();
39.         }
40.     }
41. }
```

Output:



```
iNetTutor.com
Welcome to the High-Low Number Checker!
Please enter three numbers:
56
76
24
The highest number is: 76
The lowest number is: 24
Press any key to exit.
```

Code Explanation

Line 9-11

The **Console.WriteLine()** method is used to output text to the console window. In this case, we have three separate **Console.WriteLine()** statements that display different messages.

The line **Console.WriteLine("iNetTutor.com");** outputs the text "iNetTutor.com" to the console. This message could serve as a branding or attribution line, indicating the source or organization behind the program.

The line **Console.WriteLine("Welcome to the High-Low Number Checker!");** displays the message "Welcome to the High-Low Number Checker!" in the console. This message serves as a friendly greeting or introduction to the program, providing information about its purpose or functionality.

The line **Console.WriteLine("Please enter three numbers:");** prompts the user to enter three numbers. This message instructs the user on what action to take and specifies the required input.

Line 14-16

The **Console.ReadLine()** method is used to read a line of text entered by the user in the console. In this code snippet, we have three separate lines, each assigning the user input to a respective string variable (input1, input2, and input3).

When the program execution reaches these lines, the console waits for the user to enter a line of text. The user can input any text, including numbers, letters, or symbols, followed by pressing the Enter key.

Once the user enters their input and presses Enter, the **Console.ReadLine()** method captures the entire line of text as a string. It then assigns that string value to the corresponding variable (input1, input2, or input3).

These variables (input1, input2, and input3) hold the user's input as strings, allowing you to further process or manipulate the input in subsequent code.

Line 19-22

The **int.TryParse()** method is used to attempt the conversion of a string to an integer. It takes two parameters: the string to convert and an out parameter that will hold the converted integer value if the conversion is successful. It returns a boolean value indicating whether the conversion was successful or not.

In this case, we have three separate **int.TryParse()** statements, each converting a user input string (**input1**, **input2**, **input3**) to an integer value (**number1**, **number2**, **number3**), respectively. The method attempts to parse the string, and if successful, assigns the converted integer value to the corresponding variable.

Additionally, the code includes three **boolean** variables (**isNumber1Valid**, **isNumber2Valid**, **isNumber3Valid**) that store the result of the conversion operation. These variables will be used later to check if the conversions were successful or not.

By using **int.TryParse(),** we can handle scenarios where the user enters invalid input that cannot be converted to an integer. Instead of causing an error or exception, the method will return false, indicating that the conversion failed. This allows us to handle invalid input gracefully and provide appropriate error messages or logic.

Line 24-35

The if statement is used to check if the conversions from the previous code snippet were successful (isNumber1Valid, isNumber2Valid, and isNumber3Valid are all true). This condition ensures that the user entered valid numbers that could be converted to integers.

If the condition evaluates to true, meaning all three numbers are valid, the code within the if block will execute.

Within the if block, the **Math.Max()** and **Math.Min()** methods are used to determine the highest and lowest numbers, respectively. By nesting the Math.Max() and Math.Min() functions, we compare all three numbers to find the overall highest and lowest values.

The highest number is calculated by taking the maximum of number1, number2, and number3, while the lowest number is calculated by taking the minimum of the same three values.

Then, using **Console.WriteLine(),** the program displays the highest and lowest numbers to the console, using string interpolation **($"{highest}"** and **"{lowest}")** to include the values within the output messages.

If the condition in the if statement evaluates to false, meaning at least one of the numbers was not valid, the code within the else block will execute. In this case, an error message is displayed, indicating that the input was invalid and the user should enter valid numbers.

Line 37-38

The **Console.WriteLine()** method is used to output the message "Press any key to exit." to the console. This message informs the user that they need to press a key to exit the program.

After displaying the message, the program waits for the user to press any key by using the **Console.ReadKey()** method.

Summary

In the blog post on "High Low Number in C#", we explored the concept of creating a program that determines the highest and lowest number among a user's input. We discussed the importance of such a program in various scenarios, such as data analysis or finding extremes in a dataset.

The post provided a step-by-step tutorial on building the High Low Number Checker in C#. We covered topics like capturing user input, converting strings to integers, and implementing logic to compare and identify the highest and lowest numbers.

The post emphasized the significance of error handling when dealing with user input, ensuring that the program gracefully handles invalid entries. Additionally, we highlighted the usefulness of control flow structures like if-else statements to execute specific code blocks based on certain conditions.

By the end of the tutorial, readers gained a solid understanding of how to implement a High Low Number Checker in C#, enabling them to create similar programs for their own projects or applications.